PAPER Special Section on Picture Coding and Image Media Processing

Robust Index Code to Distribute Digital Images and Digital Contents Together

Minsu KIM[†a)], **Kunwoo LEE**[†b)], ***Nonmembers***, **Katsuhiko GONDOW**[†], ***Member***, ***and*** **Jun-ichi IMURA**[†], ***Nonmember***

SUMMARY The main purpose of Codemark is to distribute digital contents using offline media. Due to the main purpose of Codemark, Codemark cannot be used on digital images. It has high robustness on only printed images. This paper presents a new color code called Robust Index Code (RIC for short), which has high robustness on JPEG Compression and Resize targeting digital images. RIC embeds a remote database index to digital images so that users can reach to any digital contents. Experimental results, using our implemented RIC encoder and decoder, have shown high robustness on JPEG Comp. and Resize of the proposed codemark. The embedded database indexes can be extracted 100% on compressed images to 30%. In conclusion, it is able to store all the type of digital products by embedding indexes into digital images to access database, which means it makes a Superdistribution system with digital images realized. Therefore RIC has the potential for new Internet image services, since all the images encoded by RIC are possible to access original products anywhere.

key words: *bar codes, robustness, digital images, image databases, image color analysis*

1. Introduction

Digital media content distribution has achieved rapid growth recently [1]. The contents are distributed in various ways, such as 2D Barcodes [2], Watermark [3], Digital Rights Management (DRM) [1], [4] and so on. However, the platform-independent distribution system being able to share digital contents on the Internet with high robustness is not realized. In [2], the contents are distributed by device-to-device transmission, so that it is not an appropriate method for sharing them. In [3], Reversible Data Hiding (RDH) using the watermark scheme is proposed but the embedded data get damaged on JPEG Compression. Due to the above reasons, DRM is mainly used to distribute contents [1], [4]. It encrypts original big contents and decrypts them on the client side. But the contents are distributed by a linked url, because the encrypted contents cannot be recovered from the compression damage of the Internet services.

To distribute digital contents with high robustness from the damage, we focus on developing an image codemark to link a JPEG image with digital contents on the Internet. JPEG image is one of the most popular media format. Despite the development of the Internet and the computer, JPEG is still the most popular choices as an image format. Because it provides the best compromise between image quality and file size, 73.5% of the Internet services use JPEG images [5]. Hence, because it is easy to share a JPEG image, linking the image with digital contents make possible to share digital contents easily on Facebook, Twitter and Messenger.

At present several codemarks targeting printed images such as barcode, Quick Response Code (QR Code) and Color Quick Response Code (CQR Code) [6] are in use. The existing codemarks make connection of digital contents with printed advertisements. It has high robustness on the offline world or the device to device transmission [2]. However, when codemark embedded images are uploaded typically on Facebook, they are resized and compressed for the fast response. For example, a 1280px-width JPEG image which has 232KB, is compressed to 35KB automatically by uploading it to Facebook [7]. On this stage, embedded data into the image by a codemark get damaged, which means the data cannot be extracted correctly.

On the other hand, Digital watermarking, a method to protect the copyrights of multimedia objects, is also a data embedding technique into digital images. Its robustness is very high compared to QR Code on the digital world, but it is still not guaranteed that embedded images can be 100% extracted by digital watermarks [8].

In this paper, revised Robust Index Code (called RIC) of our previous paper in SIGMAP 2016 [9], a brand new color codemark targeting digital images on the Internet is developed. This technique provides extremely high robustness on JPEG Compression and Resize which makes it possible to guarantee to extract 100% original embedded data on the sharing images on the Internet. The concept of RIC is shown in Fig. 1. Digital contents' owners embed their contents such as text, link, music, video, Virtual Reality (VR) contents into a cover image by RIC Encoder. RIC Encoder stores the contents in a remote database and embeds the generated database index into a cover image. With changing to small embedding capacity, not embedding real contents but embedding the index to link with them, it is possible to implement high robustness on the image degradation. Hence, RIC links a cover image with multiformed and unlimited digital contents safely. We implemented the RIC library and evaluated its performance through extensive experiments under a range of settings on specific parameters. The result shows that the data of RIC are extracted

Manuscript received December 12, 2017.
Manuscript revised April 9, 2018.
Manuscript publicized June 20, 2018.
[†]The authors are with Tokyo Institute of Technology, Tokyo, 152–0033 Japan.
a) E-mail: ross@sde.cs.titech.ac.jp (Corresponding author)
b) E-mail: kw.lee@pulit.jp (Corresponding author)
DOI: 10.1587/transinf.2017PCP0004

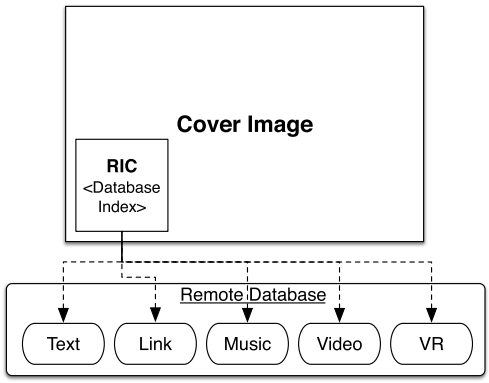

Fig. 1 RIC Concept: RIC embeds a database index, and the database stores various data types [9].

100% on the images, damaged by multiple times of over 420 pixels Resize and 30% JPEG Compression at the same time.

Moreover, we can expect a new Superdistribution system [10] based on RIC which is not restricted by any existing or new Internet platforms. Because the RIC image is platform-independent, anyone can copy and share it in all the ways, such as E-mail, SNS or mobile messenger. In addition, the RIC library supporting multiple platforms will be offered for free. Hence, it is possible that any users instinctively access to the embedded data in the RIC images very easily without restricting platforms. It is highly expected that the Superdistribution system by RIC be one of the fastest and strongest distribution routes of digital products and communication, regardless of platforms on the Internet.

Some expected examples are cited as follows. For instance, many talented artists may embed their contents and information of themselves, and retail shops may embed the links for their e-commerce system or promotion information into the their attractive images to collect traffic on the Internet. The media is also easily expected to embed the recent issues in the intriguing pictures to attract people. In case of software companies, the relatively big size contents such as games, VR contents, and so on are expected. Subsequently, those digital products embedded in the RIC images spread out by the honest needs and judgment of typical users to personally consume them on the Internet, not on the existing platforms. It means the quality and the prices of the products are only the considerations of selection and purchase without any existing interference and it is a relatively desirable system for both owners and recipients. Ultimately, this proposal aims to improve the rights of product owners and consumers, by the renovation of a distribution process and profit structure on the Internet.

2. Related Work

There are three types of data embedding methods into images. First one is Codemark, such as QR Code [11], Color Quick Response Code (CQR Code) [6], Microsoft's High Capacity Color Barcode (HCCB) [12], etc. The papers [13]–[16] provide high error correctable 2D codes and [17], [18] provide high quality visual QR codes. The main difference among the presented codes is the limited data size. Because they are targeting on mainly off-line printed images, the capacity of data is one of the main challenges of them [13]. In addition, the finder pattern boxes on existing Codemark can be easily damaged by Resize. Therefore, in spite of high recognition rate on printed images, the data embedded by the presented codes get damaged on digital image degradation such as Resize, JPEG Compression. They are not suitable for digital images on the Internet.

Second one is Digital watermark, which has high robustness on image degradation. The data embedded by watermarking algorithms are not damaged by JPEG Compression, Gaussian noise, Cropping and Resize [19]–[21]. But due to the purpose of digital watermark which is ownership evidence or fingerprinting, the watermark detection does not succeed 100% [19], [20], which means that everyone who downloads an embedded image cannot see the same data. The paper [22] tries to embed indexes into images using digital watermark. Despite its large encoded size, data loss occurs by JPEG Compression. Steganography similar to Digital watermark has high robustness too, but also embedded data can not be detected fully [23], [24].

The final one is Reversible Data Hiding (RDH). There are lots of methods on RDH, still none of them is not practical on image degradation. In [25], authors proposed a new RDH scheme with histogram modification using DC coefficients. They focus on high visual quality and small increasing file size, but they did not consider low JPEG Compression quality factor. In [26]–[28], authors proposed an image encryption method with embedding public key or original image's data into the encrypted image, but they did not consider image degradation. Image histogram is used in [29] and Modification of Prediction Errors (MPE) is used in [30] for RDH, but none of them experiments for the robustness.

To embed contents into an image and share it online, targeting digital images and 100% high robust detection are necessary. Unlike the existing Codemark, Digital watermark and RDH technologies, our proposed method RIC satisfies both requirements.

3. Proposed Approach

Our approach is a brand new color code, named Robust Index Code (RIC). RIC is a high robustness index code targeting digital images on the Internet. When a user download an image from the Internet, there is high possibility that the image got damaged. Generally, Internet services attack a digital image by Resize and JPEG Compression for the fast response on the service without user's recognition. Hence, RIC must be always detected on any Resize or JPEG Compression degradation and only targets them. To maintain high robustness of embedded data into images, RIC embeds not real digital products but a database index, which make it possible to use large areas for embedding one bit data. In

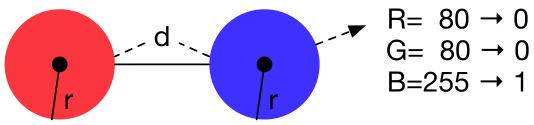

Fig. 2 Example of RIC circle [9]

addition, unlimited data can be linked with digital images because of using a database index. For now, RIC embeds only 50-bit index into each digital image, assuming a server on the Internet to store its content data. About 350 million images are uploaded to Facebook every day [31]. So 50-bit index is reasonably enough for index size.

Our concept is to design a new color code which has high robustness so that we can embed an index key of a remote database which stores lots of data. Hence, an encoded image should be able to be decoded on every platform. For this reason, we developed an encoder / decoder prototype library by C++ using OpenCV Framework [32] to deploy it to iOS, Android and Chrome Extension. Using a cell phone or Chrome browser on PC, users can easily and directly see the contents of an encoded image on the Internet. There is an example of the RIC library, which is explained in Appendix.

RIC consists of encoding, decoding and validation parts. This section describes encoding part with the design of RIC, decoding part with a new algorithm and validation part.

3.1 Robust Index Code Design

Generally, when digital images are compressed by JPEG Compression, image pixels are infected by surrounding pixels, due to the quantization step of the JPEG Compression [33]. This is the main reason why QR Code gets damaged on digital images. To solve this problem, a data pixel should be larger than other codes and the distance between data pixels should be long enough not to be infected by neighbor pixels. Hence, we propose a circle of a fixed size, which radius is r pixel for inserting bit data. Also the distance between two circles is longer than a constant d pixel which is shown in Fig. 2. The idea of embedded spot is similar to CQR Code [6], which is 24-bit RGB color. On each circle, 3 bits data can be embedded by 1 bit to R, 1 bit to G and 1 bit to B with converting 0 to color D_0 and 1 to color D_1. Figure 2 is an example of setting $D_0 = 80$ and $D_1 = 255$.

There is another infection on JPEG Compression by surrounding image pixels. Therefore, in order to protect data pixels from image pixels and detect an encoded area on image, RIC includes a square wrapping data circles. The square's background color is light gray color for not infecting data circles. For more accurate detection, the border of square is added with light black color like [34]. Assuming r to 4, d to 2, D_0 to 80 and D_1 to 255, the final design is shown in Fig. 3 used in encoding, decoding and evaluating.

From our design, the capacity of RIC is shown in the following formula:

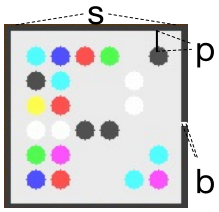

Fig. 3 Example pattern of RIC [9]

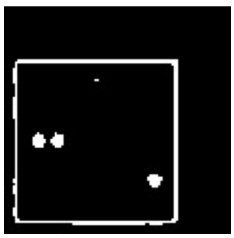

Fig. 4 Low-pass filtered RIC

$$capacity(\text{bit}) = 3N_c \ s.t.\ 0 < N_c \leq (\frac{s+d-2p}{2r+d})^2$$

where s is a side pixel of the square without the border, p is padding pixels of the square (assumed to 6) and N_c is the number of circles. Figure 3 contains 22 circles, for the design reason to represent **P** shape, so that 66 bits can be embedded. It is enough capacity to embed database index, which size is 50 bits. On the other hand, there is no need to follow the example design. Following design rules (radius larger than r pixel, distance between data larger than d pixel), users can redesign as they want with 36 squares or 30 ellipses.

To determine RIC is correctly encoded, RIC has a checksum algorithm. On RIC, a checksum is inserted on c bits and an index is inserted on the rest of bits. The algorithm is simple that a checksum value is the remainder when an index is divided into 2^c. That means we can determine true RIC with $1/2^c$ probability. In this paper, we decide on the bit length of checksum as 16.

3.2 Robust Index Code Decoder

Decoding RIC consists of three main parts, determining the data square, extracting bit data on data circles with damage check, and validating extracted data. The progress of decoding is shown in Fig. 5.

3.2.1 Step 1

A data square can be determined with High-pass filter and Low-pass filter. Because the border of a square could be uneven by image degradation and circles of inside of the square, we had to implement the square detection algorithm for ourselves. The algorithm is detecting the square itself without the border and the data circles. Because the border prevents the degradation into the square, the shape of square will remain as same as the original square like shown in

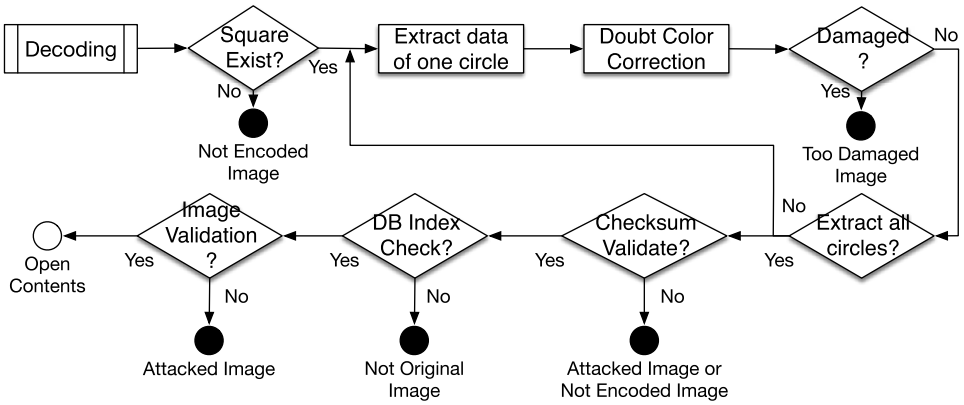

Fig. 5 Decoding process [9]

Fig. 4. In addition, the background color of the square will be black or white via High-pass or Low-pass filter. Therefore, the algorithm finds the black or white lines excluding data circles and stacks the lines. When the width of stacked lines is equal to the height of stacked lines, they can be regarded as the square. Still, when the square cannot be extracted, the image is regarded as not RIC image or extremely damaged image which means out of specification for the performance cost.

3.2.2 Step 2

Bit data is extracted from the center of data circles. Although D_0 or D_1 color circles are drawn on Encoding process, there is a high possibility that the color is not D_0 or D_1 by image degradation. Hence, doubt color correction algorithm is proposed to recover extracted data. When an extracted color from a data circle is far from D_0 and D_1, it is impossible to determine that the original bit data was 0 or 1. Therefore, the algorithm adds both 0 and 1 for that doubtable color into possible index list, called doubt list in this paper, so that actual original index is always included in the doubt list. In addition, the algorithm puts on the limit of the size of the doubt list, N_d. If the size oversteps the limit, the decoder recognizes the image as a too damaged image and stops decoding. With this algorithm, the decoder can recover damaged RIC.

3.2.3 Step 3

From the second step, the decoder extracts multiple indexes, which is the doubt list. To extract exactly only the original index, RIC decoder needs to validate the doubt list. There are three validations on RIC decoder. Firstly, validate checksum values of extracted data as we have mentioned on the encoding part. Secondly access to the remote database and validate indexes and images' ratio. In this step, RIC decoder can get the candidate images' inclination data from the extracted indexes. Finally, check image's inclination data matching up with candidate images using Histogram described in Sect. 4. At last, few validated-extracted data remains. We expect the size of the last remained doubt list is only one. If it's more than one, the user picks the right one.

4. Histogram Validation

Histogram shape is interpreted as the relative relations in the number of pixels among groups of two different bins. Image histogram shape with geometric attacks is almost invariant [35]. The idea is not new using histogram to identify a multimedia object. In [35], the authors have addressed a robust image histogram hash algorithm which generates a bitstream. This method has high robustness on identifying same images, but low uniqueness on identifying different images. In addition, it translates entire image histogram values to the hash bitstream so that the size of bitstream is very large. Because the requirements of RIC validation are small size of data, high robustness and high uniqueness to block the wrong access, this method is not suitable. In [36], the authors addressed histogram-based video hashing algorithm, but the image is not target of their method. In this paper, new image histogram hashing algorithm is proposed with high robustness and high uniqueness.

4.1 Histogram Hash

Histogram hashing on RIC, focuses on Resize and JPEG Compression for the same reason of RIC. Figure 6 shows an example of an image B channel histogram. Figure 7 shows an example of a damaged image B channel histogram attacked with 10 times of Resize and JPEG Compression. The attacked histogram shape is ruined but the x-axis value of peaks of the histogram are almost same with the original shape. Hence, uses peak data as an image identifier. There are 4 steps on generating an image identifier.

1. Divide an image into m blocks. Since the image histogram does not consider color's position, there is high possibility that an image identifier generated from a whole image itself has low uniqueness. That's why comparing divided blocks.
2. Histogram can be calculated on single channel. Separate a block into RGB channels.

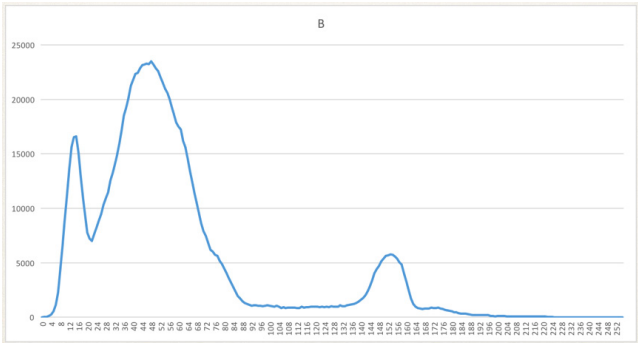

Fig. 6 Example of a B channel image histogram

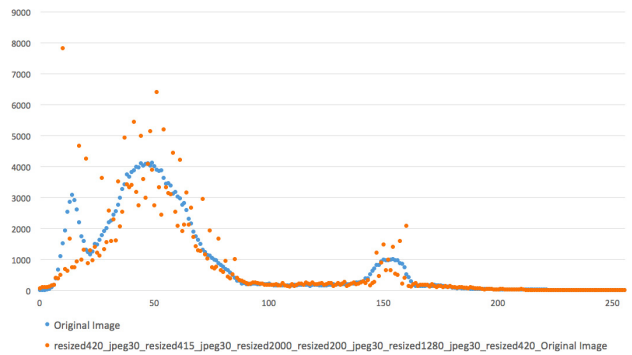

Fig. 7 Comparison of an image histogram with a damaged image histogram. Blue: an original image histogram, Orange: a damaged image histogram with 6 times of Resize and 4 times of JPEG Compression.

3. Apply Histogram Equalization on every channels from Step 2. Stressing image's characteristics make robustness high.
4. Calculate a histogram graph and pick n peaks from the biggest y-axis value. The x-axis values of n peaks on each block is the identifier of an image.

4.2 Histogram Comparison

RIC saves original products on a remote database so that we can save an image identifier based on the histogram on the Internet. Therefore, on the encoding process, RIC saves original products with validation data and embeds the generated index into the image. On the decoding process, when the indexes are extracted, our system fetches histogram identifiers from the database and compare them with decoding images. There are 4 steps on histogram identifier comparison.

1. Extract an index from a damaged RIC image and fetch a histogram identifier (I_1) from the remote database.
2. Generate a histogram identifier (I_2) from a damaged RIC image.
3. On each block, calculate percentage ($P_i, 1 \leq i \leq m$) of matches x-axis value of two histogram identifiers with α error.
4. Pick the lowest percentage (P_l) on the all blocks P_i. If P_l is greater than 60% then regard two images as the same images.

5. Evaluation

The evaluation of the RIC algorithm is involved with robustness. We used a Peak Signal-to-Noise Ratio (PSNR) as a comparison parameter between a damaged image and an original image. The bigger value of PSNR implies the less damaged image and the infinity value of PSNR implies the original image [20], [37]. We experimented the prototype of RIC encoder/decoder with Fig. 3 pattern, developed by C++ using OpenCV [32]. The experiments are simulated by watching the success number of decoding attacked images with assuming r to 4, d to 2, D_0 to 80, D_1 to 255, p to 6, N_c to 22, N_d to 2^{12}. The process consists of encoding 10,000 pictures, attacking the images and decoding the attacked encoded images. We used 9,850 private images, which mainly pictured with landscapes, foods and people, and 150 downloaded images from free photo archives morgueFile [38], which width is 1280 px. Also, to check DB index in this paper, on encoding step, the original images' attributes are saved on the memory. So on decoding step, decoder checked the extracted index and image's inclination matching up the data saved above.

5.1 Comparing with Related Work

In this experiment, there are seven attack types used to compare the proposed system with QR Code [11] and Robust watermark [39]. Three methods embed a database index into an image. QR Code, which embedded a database index with the high error correction level, is embedded on an image's left-bottom area with same pixel size of RIC. On the other hand, on Robust watermark, the two images of 400px and 640px are filled with a redundant index by the DCT Based watermarking method using Luminance component. The result is regarded as success when an extracted data is the same as an embedded data. The experiment process is cited as follows. Table 1 shows the result of this experiment.

- Step 1 : Generate a random database index.
- Step 2 : Pick an image and embed the generated index to it by QR Code, Watermark and RIC.
- Step 3 : Attack embedded images by Resize or JPEG Compression.
- Step 4 : Extract the index from attacked images and check matching up with the original index.

Table 1 shows that the embedded data using QR Code or Robust watermark do not have robustness on Resize and JPEG Compression. Especially, QR Code is more sensitive with Resize attack than JPEG Comp. The position patterns of QR code can be easily damaged with specific size because of odd or even number of pixel size, such as odd divided by even. In Watermark, it is easily distorted by JPEG Comp. On the other hand, the data embedded by our proposed method is correctly extracted over the 420px. Hence,

Table 1 Comparison experiment result [9]

Attack Type	QR Code [11] Success Rates	Watermark [39] Success Rates (400px)	Watermark [39] Success Rates (640px)	Proposed Method Success Rates
No Attack	0.9700	1.0000	1.0000	1.0000
Resize to 450px	0.7421	0.9999	1.0000	1.0000
Resize to 450px × 30% JPEG Comp.	0.5291	0.0594	0.0000	1.0000
Resize to 420px	0.0926	1.0000	0.9978	1.0000
Resize to 420px × 30% JPEG Comp.	0.2723	0.0278	0.0000	1.0000
Resize to 400px	0.9416	1.0000	0.9998	1.0000
Resize to 400px × 30% JPEG Comp.	0.5469	0.0131	0.0000	0.9999

Table 2 Robustness of resize experiment result [9]

Attack Type	Nearest-neighbor interpolation	Bilinear algorithm	Bicubic algorithm	Lanczos resampling	Box sampling
Resize to 640px	1.0000	1.0000	1.0000	1.0000	1.0000
Resize to 640px × 50% JPEG Comp.	1.0000	1.0000	1.0000	1.0000	1.0000
Resize to 640px × 30% JPEG Comp.	1.0000	1.0000	1.0000	1.0000	1.0000
Resize to 640px × 30% JPEG Comp. × Resize to 420px	1.0000	1.0000	1.0000	1.0000	1.0000
Resize to 640px × 30% JPEG Comp. × Resize to 420px × 30% JPEG Comp.	1.0000	1.0000	1.0000	1.0000	1.0000

RIC is the only image-embedding method to digital images on the Internet.

5.2 Robustness of Resize and JPEG Comp.

For evaluating RIC's robustness, the experiment includes various attack types. Basically, RIC is designed for having high robustness on the situation, which the user shares a RIC image on the Internet. Common Internet services compress images with JPEG Compression and Resize [19] because their compression methods are not noticeable to the users. Hence, RIC is targeting only Resize and JPEG Compression.

To prove RIC's robustness on Resize, we executed an experiment on various resize algorithms. There are 11 algorithms on image resizing [40] and 5 implementations on OpenCV [41]: Nearest-neighbor interpolation, Bilinear and bicubic algorithms, Lanczos resampling and Box sampling. Table 2 shows the result of this Resize experiment. On the all Resize algorithms, all the detection of RIC data succeeded.

On the JPEG Compression experiment, combining Resize with JPEG Compression is considered. Hence, more complex robustness experiment has been held with experiment cases on Table 3. Figure 8 shows a sample RIC of experiment case No.9 on Table 3. Table 4 shows the result of experiment cases.

In the experiment cases, RIC of images over 420px Resize were detected successfully. Also, the size average of doubt list was converged to 1, which proves that RIC decoder detects mostly only one original data with the doubt color correction algorithm. On the performance evaluation of RIC decoder, it took less than one second. Table 3 and 4 show that decoding time is in direct proportion to image's size and degradation.

Table 3 Robustness experiment case [9]

No	Attack Type
1	No Attack
2	50% JPEG Compression
3	30% JPEG Compression
4	Resize to 2000px
5	Resize to 2000px × 50% JPEG Compression
6	Resize to 2000px × 30% JPEG Compression
7	Resize to 420px
8	Resize to 420px × 50% JPEG Compression
9	Resize to 420px × 30% JPEG Compression
10	No. 9 × Resize to 1280px × 30% JPEG Compression
11	No. 10 × Resize to 450px
12	No. 11 × 30% JPEG Compression
13	No. 12 × Resize to 2000px
14	No. 13 × 30% JPEG Compression
15	No. 14 × Resize to 400px
16	No. 15 × Resize to 420px

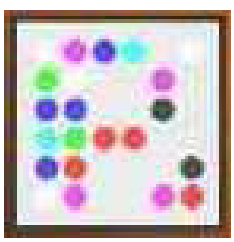

Fig. 8 Resize 420px and 30% JPEG Compression of Fig. 3. [9]

5.3 Histogram Validation

First of all, the robustness and uniqueness of the histogram identifier are experimented. 6 attack types were used to evaluate them: Resize to 700 px, Resize to 700 px × JPEG 50%

Table 4 Robustness of resize and JPEG comp. experiment result [9]

No	RIC Size(px)	PSNR Avg	Success Rates	Size Avg of Doubt List	Decode Time Avg
1	204×205	∞	1.0000	1.0000	0.0630s
2	204×205	33.76	1.0000	1.0000	0.0562s
3	204×205	32.22	1.0000	1.0000	0.0572s
4	313×314	42.33	1.0000	1.0000	0.0810s
5	313×314	37.58	1.0000	1.0000	0.0781s
6	313×314	35.81	1.0000	1.0000	0.0715s
7	73×74	31.74	1.0000	1.0000	0.0176s
8	73×74	29.47	1.0000	1.0000	0.0649s
9	73×74	28.64	1.0000	1.0000	0.4000s
10	204×205	27.71	1.0000	1.0000	0.4200s
11	77×78	27.50	1.0000	1.0000	0.4301s
12	77×78	27.47	1.0000	1.0000	0.4152s
13	313×314	27.22	1.0000	1.0000	0.4872s
14	313×314	27.10	1.0000	1.0000	0.4565s
15	70×71	28.02	0.9999	1.0000	0.4210s
16	73×74	27.90	0.9998	1.0000	0.7228s

Table 5 Histogram validation with decoding

No	Attack Type	PSNR Avg	Decoding Success Rates	Decoding Wrong Rates	Histogram Success Rates of Decoding Success	Histogram Success Rates of Decoding Wrong
1	Resize to 420px	34.30	1.0000	0.0000	1.0000	0.0000
2	No. 1 × 50% JPEG Compression	30.71	1.0000	0.0000	1.0000	0.0000
3	No. 1 × 30% JPEG Compression	29.71	1.0000	0.0000	1.0000	0.0000
4	No. 3 × Resize to 1280px × 30% JPEG Compression	29.46	1.0000	0.0000	1.0000	0.0000
5	No. 4 × Resize to 380px	29.17	1.0000	0.0000	1.0000	0.0000
6	No. 5 × 30% JPEG Compression	28.15	1.0000	0.0000	1.0000	0.0000
7	No. 6 × Resize to 350px	27.93	0.9997	0.0003	0.9998	0.0000
8	No. 7 × 30% JPEG Compression	27.21	0.9973	0.0027	1.0000	0.2222

Comp., Resize to 700 px × JPEG 30% Comp., Resize to 400 px, Resize to 400 px × JPEG 50% Comp., Resize to 400px × JPEG 30% Comp. Hence, 60,000 images were compared. We used 16 as m (the number of blocks), and 20 as n (the number of peaks). The process of evaluation is as follows.

- Step 1 : Generate a random index.
- Step 2 : Embed the generated indexes into 10,000 images by RIC.
- Step 3 : Damage RIC images with 6 attack types.
- Step 4 : In robustness experiment, compare a damaged image with the same original image. On the other hand, in uniqueness experiment, compare a damaged image with 10 different images.

It was observed that the average of identity rates in robustness was 99.93% and the average of identity rates in uniqueness was 6.020%, which means the identifier is highly unique because it recognized different images as different images. This proves that the performance of our histogram validation is very powerful with both high robustness and uniqueness. However, true-negative cases still exist. RIC don't accept a validation process which results true-negative not to remove the true index from the doubt list. Therefore, we need to improve our histogram validation.

Secondly, histogram validation with the decoding is experimented. After the decoding and existing validation processes, one or few doubt indexes will be given. Validates them with the proposed histogram validation so to remain only the original index.

Table 5 shows the result of the histogram validation with decoding experiment. On No. 7 attack type, wrong indexes extracted from 3 images because of extremely small size images. Without the proposed validation method, different data linked with the wrong index will be shown to the user. On the other hand, the wrong approach can be prevented with the histogram validation. Histogram Success Rates of Decoding Wrong from the Table 5 is 0, which means none of wrong indexes passed the validation so that it proves the validation method works properly. However, on No. 8 attack type, histogram success rates of wrong indexes is nearly 20%, because the color distribution on the image is normalized when the image is extremely small.

6. Superdistribution System

Superdistribution system [10] is an ideal proposal to distribute digital products in which software is made available freely and without restriction but is protected from modi-

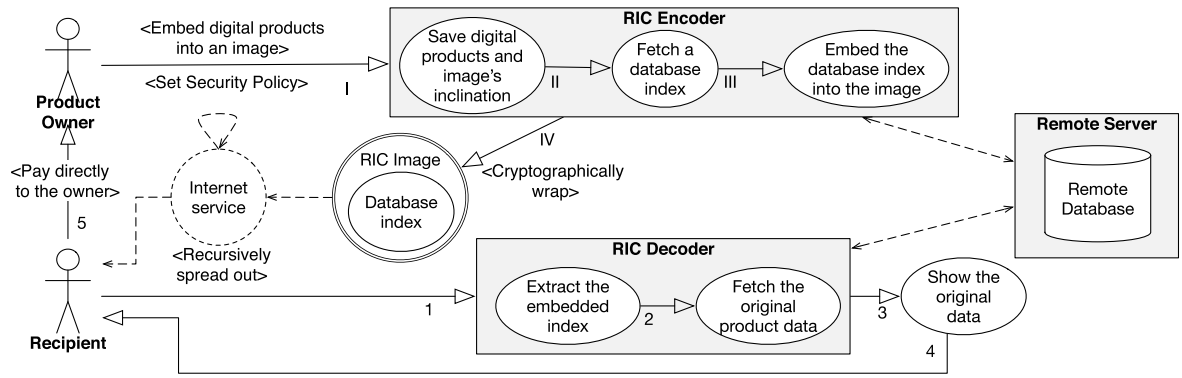

Fig. 9 New superdistribution system architecture [9]

fications and modes of usage not authorized by its vendor, and normally there are three principal functions [42] :

- A cryptographic wrapper for digital products that cannot be removed and remains in place whenever the product is copied.
- A digital rights management system for tracking usage of the product and assuring that any usage of the product or access to its code conforms to the terms set by the product owner.
- An arrangement for secure payments from the product's users to its owner.

The idea above is the most optimized architecture for both product owners and recipients for the reason that there is no restraint or additional expense through several distribution channels. Unfortunately, the complete concept of Superdistribution systems barely exists in the reality, but there are several partial Superdistribution systems which intend to make a massive profit from the distribution process on the Internet. In other words, massive platform companies have built up their own ecosystem to make a profit from both product owners and recipients. For example, Apple app store is a huge cash cow for Apple taking a flat 30% cut of all App Store transactions, earning Apple over 10 billions USD in revenue in a year [43].

To the best of our knowledge, there has not been a complete Superdistribution system. To design the desired system, we need the new source as a cryptographic wrapper which is able to embed safely both digital products and security, payment arrangement from a product owner, and barely damaged by all the expected degradation on the Internet, and also recursively spread out through Internet services.

For the reasons above, we considered three basic conditions for the wrapper of the new Superdistribution system. The digital image has received significant attention for the first condition, and the concept of RIC has been designed for the second and third condition in the first place.

- Intriguing sources for recipients, which spread out easily, regardless of platforms
- An almost unlimited capacity for diverse types of contents, and low risk for security issues
- Extremely high robustness against degradation on distribution process on the Internet

Product owners embed their digital products, security or payment policy into a digital image by RIC Encoder. RIC Encoder preferentially extracts inclination which shows the features of the digital image, such as Histogram, Ratio and so on, and save them with digital products into a remote database. Subsequently, the index of the address to access the data is embedded on the image by RIC. The new embedded image is called RIC Image, and used as a cryptographic wrapper in this architecture, shown in Fig. 9.

The RIC image is exposed to recipients by being recursively copied and spreading out on the Internet by existing Internet services like SNS. Afterwards the recipients are able to extract the products, security or payment policy of product owners easily, if RIC Decoder Library is implemented in their devices. Moreover, even if a codemark is deleted from an image, the original products remain on a remote database so that recipients are able to access the data from other RIC images.

Compared to the existing distribution process, the main difference is that all the data which product owners intend can be easily distributed, regardless any existing platforms, and recipients consume the data by the policy arranged from the product owner, not platformers, because the image contains the products. For instance, talented artists embed their music contents and concert information of themselves into the their attractive images, which means they get RIC images linked with their contents. The thing they only have to do for advertising themselves is uploading RIC images to SNS like Facebook, Twitter, LINE, Instagram, Pinterest, WhatsApp and so on. Users who find the RIC image can easily access to the original contents like right clicking the image on the Google Image Search on PC. Hence, RIC is the only method targeting digital images on the Internet so that the new proposed Superdistribution system can be only realized by RIC. When the RIC library becomes a middleware on the Internet construction, the proposed Superdistribution system will be realized.

7. Conclusion

This paper presented the design and implementation of the

extremely high robustness codemark technology, named RIC. We have determined that a digital image is one of the most desirable sources, for reasons that users already get used to use it for communication, and it is relatively easy to be copied and shared. RIC has the apparent advantage to link a digital image with unlimited digital data, and also guarantees most cases of degradation occurred on the Internet, so that the strengths of digital images above can be maximized. Our experiments validate the robustness of RIC.

To our knowledge, there is no digital image service using embedding data technologies into images. RIC is mainly designed for digital images for challenging to open a new Superdistribution system, compared to the existing ecosystems provided by platform companies. Using RIC, it is possible that RIC images can spread out in any sharing way, such as E-mail, SNS, chat-app, and the original embedded digital data can be easily extracted by end recipients on any Internet services, regardless of platforms. We had already implemented RIC to iOS, Android, Chrome browser extension, and so on to confirm that RIC can be operated across diverse platforms.

In spite that high robustness of RIC is realized, the performance of RIC, such as extracting time is not completely considered. For example, the doubt color correction algorithm helps detect damaged images, but it is highly likely that decoding process of terribly damaged images takes relatively long time when N_d is big. We keep trying the way of optimizing a doubt color correction algorithm with a small value of N_d. In addition, research for image's inclination matching with the histogram is not enough, and we are still trying several additional image matching features. These future works will upgrade the completeness and stability of the system, which also means RIC will possibly be able to be operated in the worse conditions than the currently defined specification.

References

[1] P. Korus, J. Białas, and A. Dziech, "Towards practical self-embedding for JPEG-compressed digital images," IEEE Trans. Multimedia, vol.17, no.2, pp.157–170, Feb. 2015.
[2] A. Motahari and M. Adjouadi, "Barcode Modulation Method for Data Transmission in Mobile Devices," IEEE Trans. Multimedia, vol.17, no.1, pp.118–127, Jan. 2015.
[3] A. Briassouli, P. Tsakalides, and A. Stouraitis, "Hidden messages in heavy-tails: DCT-domain watermark detection using alpha-stable models," IEEE Trans. Multimedia, vol.7, no.4, pp.700–715, Aug. 2005.
[4] T. Thomas, S. Emmanuel, A.V. Subramanyam, and M.S. Kankanhalli, "Joint Watermarking Scheme for Multiparty Multilevel DRM Architecture," IEEE Trans. Inf. Forensics Security, vol.4, no.4, pp.758–767, Dec. 2009.
[5] W3Techs, "Usage of image file formats for websites." [Online]. Available: http://w3techs.com/technologies/overview/image_format/all, 2015.
[6] K. Antoni, H. Nurwono, and R. Kosala, "Color Quick Response Code for Mobile Content Distribution," Proc. 7th Int. Conf. Advances in Mobile Computing and Multimedia, MoMM '09, New York, USA, pp.267–271, ACM, Dec. 2009.
[7] PhotoShelter, "Why Facebook Photos Look so Bad, and the DIY Solution to Fix It." [Online]. Available: http://blog.photoshelter.com/2014/01/facebook-photos-look-bad-diy-solution-fix/, 2014.
[8] S.P. Mohanty and B.K. Bhargava, "Invisible Watermarking Based on Creation and Robust Insertion-extraction of Image Adaptive Watermarks," ACM Trans. Multimedia Comput. Commun. Appl., vol.5, no.2, pp.12:1–12:22, Nov. 2008.
[9] M. Kim, K. Lee, K. Gondow, and J.-I. Imura, "Robust Index Code with Digital Images on the Internet," Proc. 13th Int. Joint Conf. e-Business and Telecommunications, ICETE 2016, pp.28–37, July 2016.
[10] R. Mori and M. Kawahara, "Superdistribution: The Concept and the Architecture," IEICE Trans., vol.73, no.7, pp.1133–1146, July 1990.
[11] D.W. Incorporated., "About QR Code." [Online]. Available: http://www.qrcode.com/, 2008.
[12] M. Research, "High Capacity Color Barcodes." [Online]. Available: http://research.microsoft.com/en-us/projects/hccb/, 2010.
[13] A. Grillo, A. Lentini, M. Querini, and G.F. Italiano, "High Capacity Colored Two Dimensional codes," Proc. Int. Multiconference on Computer Science and Information Technology (IMCSIT), pp.709–716, Oct. 2010.
[14] T. Hao, R. Zhou, and G. Xing, "COBRA: Color Barcode Streaming for Smartphone Systems," Proc. 10th Int. Conf. Mobile Systems, Applications, and Services, MobiSys '12, pp.85–98, ACM, June 2012.
[15] X. Liu, D. Doermann, and H. Li, "A Camera-based Mobile Data Channel: Capacity and Analysis," Proc. 16th ACM Int. Conf. Multimedia, MM '08, pp.359–368, ACM, Oct. 2008.
[16] C.T. Yeh and L.H. Chen, "A system for a new two-dimensional code: Secure 2D code," Machine Vision and Applications, vol.11, no.2, pp.74–82, Oct. 1998.
[17] Y.H. Lin, Y.P. Chang, and J.L. Wu, "Appearance-Based QR Code Beautifier," IEEE Trans. Multimedia, vol.15, no.8, pp.2198–2207, Dec. 2013.
[18] S.S. Lin, M.C. Hu, C.H. Lee, and T.Y. Lee, "Efficient QR Code Beautification With High Quality Visual Content," IEEE Trans. Multimedia, vol.17, no.9, pp.1515–1524, Sept. 2015.
[19] C.S. Woo, J. Du, and B. Pham, "Performance Factors Analysis of a Wavelet-based Watermarking Method," Proc. 2005 Australasian Workshop on Grid Computing and e-Research - Volume 44, ACSW Frontiers '05, pp.89–97, Australian Computer Society, Inc., Jan. 2005.
[20] M. Jiansheng, L. Sukang, and T. Xiaomei, "A Digital Watermarking Algorithm Based On DCT and DWT," Proc. Int. Symp. Web Information Systems and Applications, WISA '09, pp.104–107, Academy Publisher, May 2009.
[21] C. Wang, J. Ni, J. Huang, R. Zhang, and M. Huang, "Robust and High Capacity Image Watermarking Based on Jointly Coding and Embedding Optimization," Information Hiding, Lect. Notes Comput. Sci., vol.4567, pp.65–79, Springer Berlin Heidelberg, June 2007.
[22] J. Jiang and A. Armstrong, "Data hiding approach for efficient image indexing," Electronics Letters, vol.38, no.23, pp.1424–1425, Nov. 2002.
[23] Q. Liu, A.H. Sung, and M. Qiao, "Neighboring Joint Density-based JPEG Steganalysis," ACM Trans. Intell. Syst. Technol., vol.2, no.2, pp.16:1–16:16, Feb. 2011.
[24] N. Provos and P. Honeyman, "Detecting Steganographic Content on the Internet," tech. rep., ISOC NDSS 2002, Aug 2001.
[25] F. Huang, X. Qu, H.J. Kim, and J. Huang, "Reversible Data Hiding in JPEG Images," IEEE Trans. Circuits Syst. Video Technol., vol.26, no.9, pp.1610–1621, Sept. 2016.
[26] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and Reversible Data Hiding in Encrypted Images with Public Key Cryptography," IEEE Trans. Circuits Syst. Video Technol., vol.26, no.9, pp.1622–1631, Sept. 2016.
[27] W. Zhang, H. Wang, D. Hou, and N. Yu, "Reversible Data Hiding

in Encrypted Images by Reversible Image Transformation," IEEE Trans. Multimedia, vol.18, no.8, pp.1469–1479, Aug. 2016.
[28] Z. Qian, X. Zhang, and S. Wang, "Reversible Data Hiding in Encrypted JPEG Bitstream," IEEE Trans. Multimedia, vol.16, no.5, pp.1486–1491, Aug. 2014.
[29] W.L. Tai, C.M. Yeh, and C.C. Chang, "Reversible Data Hiding Based on Histogram Modification of Pixel Differences," IEEE Trans. Circuits Syst. Video Technol., vol.19, no.6, pp.906–910, June 2009.
[30] W. Hong, T.S. Chen, and C.W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors," J. Systems and Software, vol.82, no.11, pp.1833–1842, Nov. 2009.
[31] internet.org, "A Focus on Efficiency." [Online]. Available: http://internet.org/efficiencypaper, 2013.
[32] Itseez, "OpenCV." [Online]. Available: http://opencv.org, 2000.
[33] B. Furht, "A Survey of Multimedia Compression Techniques and Standards. Part I: JPEG Standard," Real-Time Imaging, vol.1, no.1, pp.49–67, April 1995.
[34] S.K. Ong, D. Chai, and K. Tan, "The Use of Border in Colour 2D Barcode," Int. Symp. Parallel and Distributed Processing with Applications, ISPA '08, pp.999–1005, Dec. 2008.
[35] S. Xiang, H.J. Kim, and J. Huang, "Histogram-based image hashing scheme robust against geometric deformations," MM&Sec, pp.121–128, Sept. 2007.
[36] K. Hamon, M. Schmucker, and X. Zhou, "Histogram-Based Perceptual Hashing for Minimally Changing Video Sequences," 2006 Second Int. Conf. on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06), pp.236–241, Dec. 2006.
[37] A. Hore and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 20th Int. Conf. Pattern Recognit. (ICPR), pp.2366–2369, Aug. 2010.
[38] K. Connors, M. Connors, and J. Seemann, "morgueFile." [Online]. Available: http://www.morguefile.com, 1996.
[39] M. Yesilyurt, Y. Yalman, and A.T. Ozcerit, "A New DCT Based Watermarking Method Using Luminance Component," Electronics & Electrical Engineering, vol.19, no.4, pp.47–52, April 2013.
[40] Wikipedia, "Image scaling." [Online]. Available: https://en.wikipedia.org/wiki/Image_scaling, 2016.
[41] OpenCV, "Geometric Image Transformations." [Online]. Availalbe: http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#resize, 2016.
[42] Wikipedia, "Superdistribution." [Online]. Available: https://en.wikipedia.org/wiki/Superdistribution, 2013.
[43] Appleinsider, "Apple's App Store generated over $10 billion in revenue for developers in record 2014." [Online]. Available: http://appleinsider.com/articles/15/01/08/apples-app-store-generated-over-10-billion-in-revenue-for-developers-in-record-2014, 2014.

Appendix A: Example of the RIC library

This appendix describes an example of the prototype iOS application of the RIC library. Figure A· 1 shows a RIC image linked with the food information. When users open the image in the app, food name, description, location and so on will be displayed in Fig. A· 2.

Fig. A· 1 Example of the RIC image [9]

Fig. A· 2 Extracted information of the RIC image in the Application [9]

Minsu Kim received the B.S. and M.S. degrees in Computer Science from Tokyo Institute of Technology in 2013 and 2015, respectively. He is now a Ph.D. student in Tokyo Institute of Technology.

Kunwoo Lee received the M.S. degree in Mechanical and Environmental Informatics from Tokyo Institute of Technology in 2015. He is now the CEO in Pulit Inc.

Katsuhiko Gondow received the Ph.D. degree in Computer Science from Tokyo Institute of Technology in 1994. He is a professor in Computer Science of Tokyo Institute of Technology.

Jun-ichi Imura received the Ph.D. degree in Mechanical Engineering from Kyoto University in 1995. He is a professor in the Department of Mechanical and Environmental Informatics of Tokyo Institute of Technology.